Inferring functional connectivity from time-series of events in large scale network deployments

Article (Accepted Version)

Inferring Functional Connectivity from Time-series of Events in Large Scale Network Deployments

Antoine Messager*, George Parisis*, Istvan Z. Kiss†, Robert Harper‡, Phil Tee§ and Luc Berthouze*
*Department of Informatics, University of Sussex, Brighton, UK, Email: {A.Messager, G.Parisis, L.Berthouze}@sussex.ac.uk †Department of Mathematics, University of Sussex, Brighton, UK, Email: I.Z.Kiss@sussex.ac.uk ‡Moogsoft Ltd., Kingston upon Thames, UK, Email: rob@moogsoft.com §Moogsoft Inc., San Francisco, USA, Email: phil@moogsoft.com

***Abstract*—To respond rapidly and accurately to network and service outages, network operators must deal with a large number of events resulting from the interaction of various services operating on complex, heterogeneous and evolving networks. In this paper, we introduce the concept of functional connectivity as an alternative approach to monitoring those events. Commonly used in the study of brain dynamics, functional connectivity is defined in terms of the presence of statistical dependencies between nodes. Although a number of techniques exist to infer functional connectivity in brain networks, their straightforward application to commercial network deployments is severely challenged by: (a) non-stationarity of the functional connectivity, (b) sparsity of the time-series of events, and (c) absence of an explicit model describing how events propagate through the network or indeed whether they propagate. Thus, in this paper, we present a novel inference approach whereby two nodes are defined as forming a functional edge if they emit substantially more coincident or short-lagged events than would be expected if they were statistically independent. The output of the method is an undirected weighted graph, where the weight of an edge between two nodes denotes the strength of the statistical dependence between them. We develop a model of time-varying functional connectivity whose parameters are determined by maximising the model's predictive power from one time window to the next. We assess the accuracy, efficiency and scalability of our method on two real datasets of network events spanning multiple months and on synthetic data for which ground truth is available. We compare our method against both a general-purpose time-varying network inference method and network management specific causal inference technique and discuss its merits in terms of sensitivity, accuracy and, importantly, scalability.**

***Index Terms*—network management; network events; functional connectivity inference.**

I. Introduction

SWIFTLY identifying network and service outages to ensure network and service availability in modern, large-scale networks is crucial [1]. Network operators continuously collect log data from all devices and running processes that are deemed to be important. Ensuring continuous network and service availability relies on the efficient and effective analysis of collected data so that outages can be quickly identified or predicted before user experience gets disrupted. This is a very challenging task. Networks are large, complex, heterogeneous and evolving. They support diverse services that are widely distributed, and also evolving. The aggregate rate of collected events is commonly high due to the very large number of monitored devices and services; a typical rate for a large-scale network deployment would be 10^6 events per second [2]. However, although the aggregate event rate is large, the rate at which individual devices emit events is extremely low such that correlating emitted events is inherently challenging; this becomes even more cumbersome in the presence of periodical informational events [3]. In addition, the vast majority of collected event data is noise and only a few of them may correlate with actionable incidents. Concurrency across network and services results in collected events whose timestamps may be unreliable in terms of absolute values and ordering, due to misconfiguration or loose synchronisation. This makes workflow-based anomaly detection [4] and concurrent log analysis approaches [5] difficult to apply. Finally, there is no explicit model describing the precise mechanisms responsible for the generation of events in the network when an outage or a software failure occurs. For example, black holes due to routing failures may take seconds or minutes to manifest themselves, whereas application servers will start emitting error events immediately after contacting a failed authentication server. Events may not be emitted at all by a failed or failing device/service or a separate monitoring device may emit failure-related events on behalf of unreachable devices after polling a failed device and devices/services that are known to be attached to it (e.g. a server hosting Docker containers or a ToR switch connecting data centre servers), as discussed in [6]. Note that in the latter case, the event emission pattern and frequency is independent of the underlying structural connectivity and solely depends on the configuration of the external monitoring system.

Root Cause Analysis [7] has therefore been a prominent research area. Network operators commonly employ rule-based analysis where a pre-defined and manually updated list of rules is used to exclude uninteresting log data and make analysis of remaining events practical. This is a time consuming and error-prone process. Misconfiguration may result in fatal outages which could have been otherwise easily detected or predicted [8]. Kobayashi et al. [3] recently proposed an inference algorithm for mining causality of network events that has been shown to have good performance. However, the algorithm's complexity is cubic in the number of network events. In model traversing techniques one explores progressively the neighbours of each entity emitting an event to identify its source [9] using a formal representation of the network structure.

In this paper, we seek to offer a radically different take on how network management operators could go about monitoring, responding to, and even predicting, network and service outages. This new perspective relies on the concept of *functional connectivity* within the network. The term functional connectivity was coined in the field of neuroscience and refers to *"an observable phenomenon that can be quantified with measures of statistical dependencies, such as correlations, coherence, or transfer entropy"*. Importantly, such connectivity is not assumed to denote any causal influence (in which case, the terminology used is *effective connectivity*). From a network management viewpoint, a functional connectivity could therefore denote a number of different things. It could, for example, refer to the integrated involvement of a set of network nodes in the provision of a particular service[1] but it could equally represent a set of network nodes that appear to be systematically involved whenever a particular kind of hardware or software failure occurs. Functional connectivity is underpinned by, but distinct from, *structural connectivity*, a description of the actual physical network infrastructure (including end-hosts, switches, routers, firewalls, NAS devices and any other middleboxes present in the network) typically obtained by taking a dump of the customers operations database. Such database is fed by the change management and connectivity discovery systems, and is automatically updated when network links are provisioned/de-provisioned or equipment is configured.

By design, our approach is agnostic to whether an operational meaning can be readily attributed to the inferred connectivity. Rather, it is a data-driven approach that identifies nodes as having statistical dependencies between their activities. Specifically, we measure these statistical dependencies in terms of properties of the distribution of delays between the events emitted by the nodes. We develop an inference method whose output is an undirected weighted graph where the weight of an edge between two nodes denotes the strength of the statistical dependency between them. Our method does not rely on event pre-processing and de-duplication, therefore it is very efficient in terms of execution time and memory requirements. In contrast to [3], we take advantage of all events, including purely informational, periodical events, to infer functional connectivity between network nodes even in the absence of failures or service outages. Depending on the structure of the graph that is being produced (for example, if it consists of multiple connected components or features a strong modularity index), the output of our method can be interpreted in terms of one or many functional groups consisting of a number of nodes; and a node may belong to multiple functional groups (e.g. servers running different VMs supporting multiple cloud tenants' services). With our method a network operator is informed at all times about ever-changing service deployments (and the underlying network topology which can also be seen as a functional one at the physical/link or IP layers). We believe this provides a powerful tool for swiftly responding to, and investigating the root causes of recent or imminent failures, based solely on the times of events emitted by devices.

[1]Services may be realised at different layers by in-network and end-host devices; e.g., a set of routers that form an OSPF area, a set of switches that are part of a spanning tree, or an application deployment that consists of application and database servers, load balancers and a firewall.

The paper is organised as follows. In Section II, we describe the proposed methodology. In Section III, we validate the method by applying it to real-world data and quantifying its predictive power. We then benchmark it against two state-of-the-art methods on both real-world data and synthetic data for which ground truth is available. Finally, we provide results regarding scalability. Section IV discusses research related to our work. We conclude by discussing limitations and possible avenues for further work (Section V).

II. Functional Connectivity Inference

Measures of statistical dependence typically used in functional connectivity inference include correlations, coherence and transfer entropy. However, their applicability to event times of devices in large-scale network deployments is severely undermined by (a) the sparsity of events at node level and (b) the lack of knowledge as to how precisely the timing of events is being recorded, or indeed whether this timing is artificially induced by how the network management system obtains or records events (e.g., polling might be involved). Instead, in what follows, we introduce a statistic that meets those challenges and for which confidence intervals have been analytically derived, irrespective of either sparsity or duration of the data. Given two point processes X and Y, each emitting a given number of discrete events m and n on a fixed period of time T, this statistic quantifies the likelihood that the observed number of pairs of events $\left(X_i, Y_j\right)$ separated by a delay of less or equal than τ could be expected if X and Y were independent. This statistic is the basis of our assessment that two nodes are functionally connected (i.e., that they are statistically dependent). In the following subsection, we describe the statistic and how it is used in a windowed measure of the temporal relationship between the events emitted by two nodes. This measure (referred to as score thereafter) will then be used to build a model of time-varying edge probabilities (which will be described in Section II-B).

A. Score: estimating pairwise statistical dependence

The data we were provided with (described more fully in the next Section) consisted of sequences of integer event times (Unix time stamps in seconds) for each device in the network. For the purpose of our statistic, each time-series was interpreted as a fixed-length sequence of 0's and 1's where 1's denoted the presence of an event and 0's the absence of an event. Then, for each pair of nodes, we used a simple adaptation of the cross-correlation function to count the number of times their respective events occurred within less than a given lag (delay) δ of one another, making no distinction between positive and negative delays (thus partly addressing the challenge of concurrency). Formally, we calculated

$$S_{T,\delta} = \{(i,j) \in [\![1,T]\!]^2 : |i-j| \leq \delta, X_i = Y_j = 1\}. \quad (1)$$

To assess the presence of statistical dependence, we compared this quantity with its expected value and standard deviation

when X and Y are independent and identically distributed uniform random variables emitting the same number of events n_X and n_Y over the same period of time T. These values were analytically derived in [10] and summarised here:

$$\mathbb{E}(|\widetilde{S}_{T,\delta}|) = p_X p_Y\Big(T - (T-\delta))^2\Big) + p_X p_Y (T-\delta), \quad (2)$$

and

$$\sigma^2_\delta = (2\delta+1)p_X p_Y(1-p_X p_Y) + 2\delta(2\delta+1)p_X p_Y(p_Y(1-p_X)+p_X(1-p_Y)), \quad (3)$$

where parameters p_X, p_Y are set to their empirical estimators $\frac{n_X}{T}$ and $\frac{n_Y}{T}$ respectively.

The derivation [10] of a central limit theorem demonstrating convergence of the distribution of this statistic to a normal distribution of known parameters finally enables us to construct the following Z-score:

$$Z_{X,Y}(\delta) = \frac{|S_{T,\delta}| - \mathbb{E}(|\widetilde{S}_{T,\delta}|)}{\sigma_\delta \sqrt{T}} \quad (4)$$

quantifying the likelihood of X and Y being functionally connected.

Since calculating cross-correlations over all possible pairwise interactions is computationally intensive when considering a large-scale network, we typically limited ourselves to a maximum delay δ_{max} as specified in Section III-F and used the average over all lags up to δ_{max} as our final score.

B. Model of time-varying connectivity

In this section, we describe our approach to translating the scores introduced in Section II-A into time-varying probabilities of the existence of functional edges. Since scores require estimates of cross-correlations, a fundamental assumption of the method is that of separation of timescales; changes in functional connectivity should occur much slower than the rate at which processes generate events; this is a realistic assumption in the context of computer network management. Changes in the functional connectivity occur when hardware is commissioned / de-commissioned and services are deployed / un-deployed. Even in very dynamic network deployments that support elastic cloud services, changes in the functional connectivity can be safely assumed to take place at timescales that are significantly smaller than the respective event generation rates (a range of time windows will be considered in Section III). Another source of changes are failing devices (e.g., servers, routers). Such failures do happen frequently, especially in large-scale deployments, however, they result in a stream of events (by neighbouring or monitoring devices) and therefore provide information to our method about functional connectivity around the failing node.

A key principle of the proposed methodology is that the score $s_e(t_w)$ for a pair of nodes within a time window t_w provides the information required to update the estimate of the value of the probability $p_e(t_w - 1)$ of a functional edge existing between these nodes at the previous time window. More precisely, we consider that information is gained about the probability of an edge existing only when both nodes emit events during the time window considered. This is a natural implication of the sparsity constraint. The fact that only one node in a pair emits an event does not necessarily imply that an edge does not exist (or no longer exists). For each pair of nodes and each time window t_w, there are therefore three cases to consider:

1) The score is positive, $s_e(t_w) > 0$, i.e., there were more coincident or short-lagged events between these two nodes than between randomly picked pairs of nodes with similar levels of activity. This increases confidence about the existence of an edge and therefore the probability $p_e(t_w)$ should increase as some function h_1 of the score.
2) The score is negative, $s_e(t_w) \leqslant 0$, i.e., there were fewer coincident or short-lagged events than expected at random. This lowers confidence about the existence of an edge and therefore the probability $p_e(t_w)$ should decrease as some function h_2 of the score.
3) At least one of the node does not emit events: This scenario does not provide any information and the probability should remain unchanged.

This leads to the following model formulation:

$$p_e(t_w+1) = \begin{cases} \Big(1-(1-p_e(t_w)) \times (1-h_1(s_e(t_w)))\Big) & \text{if } s_e(t_w) > 0, \\ p_e(t_w)\times(1-h_2(s_e(t_w))) & \text{if } s_e(t_w) \leqslant 0, \\ p_e(t_w) & \text{if no information,} \end{cases} \quad (5)$$

If h_1 and h_2 are continuous, monotonically increasing and decreasing, respectively, functions of the score with output in $[0;1]$, this formulation ensures that $p_e(t_w)$ remains in $[0;1]$. In our implementation, h_1 and h_2 are simple sigmoid functions, each involving a single free parameter (referred to as α and β thereafter). Other formulations are possible but do not affect the principle of the method, provided they are differentiable in their parameter(s). Since changes in functional connectivity from one window to the other are assumed to be small, we formulate the problem of determining the two free parameters as one of minimising the error of a binary classifier predicting the sign of the score at time t_w given the edge probability at time $t_w - 1$. In other words, if the edge probability at time $t_w - 1$ is greater than a threshold th (0.5 throughout) and both nodes emit events in time window t_w, we expect the score at time t_w to be positive. Conversely, if the edge probability at time $t_w - 1$ is less than the threshold and both nodes emit events in time window t_w, we expect the score at time t_w to be negative. Our error criterion is formally defined as:

$$E = \frac{1}{2}\sum_{t_w=1}^{N_w}\Bigg(\sum_{s_e(t_w)\leqslant 0 \text{ and } p_e(t_w-1)\geqslant th}(p_e(t_w-1)-th) + \sum_{s_e(t_w)>0 \text{ and } p_e(t_w-1)<th}(th - p_e(t_w-1))\Bigg). \quad (6)$$

It penalises misclassifications, namely $p_e(t_w) \geqslant th$ and $s_e(t_w) \leqslant 0$, or $p_e(t_w) > th$ and $s_e(t_w < 0)$), with a cost

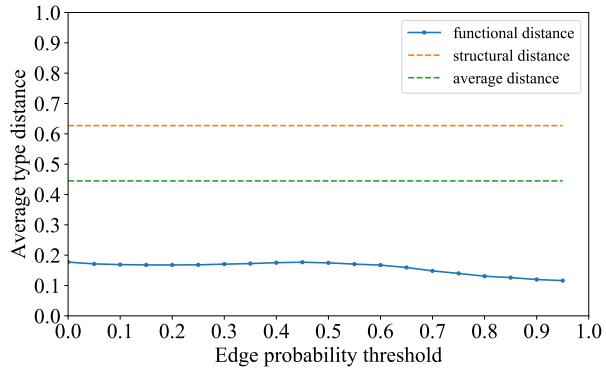

Fig. 1. Evolution of the average type distance over all connected pairs of device as a function of the probability edge threshold (blue). Average type distance between not functionally connected pair of nodes (dashed green). Average type distance between structurally connected pair of nodes (dashed orange).

proportional to the difference between edge probability and threshold. As a sum of edge probabilities that are differentiable functions of the parameters, a simple gradient descent can be used to determine the values of the free parameters. Time-varying edge probabilities can then be calculated for all pairs using the update equation (5).

III. Experimental results

A. Description of the datasets

1) Real Datasets: This paper is based on datasets of network events and underlying physical network topologies from two different organisations[2]. The first (**Dataset 1** thereafter) consists of network events and the underlying physical topology from a large retail bank. The infrastructure supports both central and distributed operations in remote sites. The network consists of a core of meshed routers, distribution switches and the supported application and infrastructure servers. The network supports both hub, hub to spoke and intra-spoke operations for financial transactions, and the supporting back office systems. Network events span a duration of 54 days (in period 5/2018 to 06/2018). Structural network topologies were obtained in the middle and at the end of the record. Overall 13,428 different nodes emitted 3,000,418 events leading to a mean value of 4.14 events per node per day. Just 1% of the nodes emitted 65% of all events whilst only 260 nodes (or 2%) emitted more than one event per hour. The second dataset (**Dataset 2** thereafter) corresponds to a Fortune 500 technology company and comprises a core of meshed backbone routers and a distribution layer of switches. It supports the company's commercial operations, including accounting, human resources, research and development, telephony and sales. Network events span a duration of five months (2/5/2015 to 25/11/2015). Considering only those 10,984 nodes in the giant component that emitted events, there were 2,189,579 events leading to a mean value of 1.36 events per node per day. More than half of the nodes emitted only up to 1 event per month whereas less than 3% of the nodes emitted more than 1 event per day. Only a tiny fraction of the nodes emitted more than 1 event per hour.

[2]These datasets are currently not publicly available due to their commercially sensitive nature.

2) Construction of the synthetic data: Since validating a method that infers *functional* connectivity is very challenging because more often than not no ground truth is available (as with our real datasets – but see Section III-B), we designed and generated a synthetic dataset capturing as many properties of the real system as possible. First, to enable sensitivity analysis over a large number of scenarios and parameters, we generated scaled-down versions of the actual structural connectivity, i.e., the actual physical network infrastructure. The generation process was as follows. Initialise a first list L_1 with a node n picked at random from the network. Initialise a second list L_2 containing the neighbours of node n. Then, until L_1 reaches the desired size, repeat: choose a member of L_2 at random with probability proportional to its number of neighbours in L_1; add to L_1; update L_2. The resultant graph is the subgraph induced by the vertices in L_1. This process guarantees that the graph generated is connected. We confirmed that this simple process approximately preserved key features of the true topology, specifically, the degree distribution and the distributions of local clustering, local assortativity and betweenness centrality. This is illustrated qualitatively by the top 4 rows of Figure 2 and quantitatively by the bottom row, using Jensen-Shannon Divergence (JSD) when varying the size of the synthetic data from 100 to 10,000 nodes. We note that whilst there are a number of methods able to generate graphs with a prescribed degree distribution (as well as a limited number of other features), we are not aware of any network generative mechanism preserving the above set of properties (whether of the same size or otherwise).

Next, we defined two classes of event-emitting processes. The first class involves functional connectivity in so far as the events are produced as in the four types of failure scenarios identified in [6]. Each scenario involved a different temporal pattern of events. To model *container failures*, in which a server failure leads to events being associated to the contained virtual machines (following probing by the management system), we organised neighbours of high-degree (10+) nodes into a number of disjoint functional groups, each node modelling a different virtual machine. For each container failure, one functional group was chosen at random and events were emitted on behalf of all nodes of that functional group approximately 30s after the failure, with some jitter allowing for bursts spanning approximately 5 seconds. In *intelligent polling failure* scenarios, a similar setup was used except that the events occurred approximately once per minute on a round-robin basis, up to some specified duration. To simulate *flapping interface* failures, we randomly picked two nodes among those nodes with the highest betweenness centrality and constructed a functional group out of all nodes located on the shortest path in the structural connectivity between them. All nodes in the group emitted events at a rate of one every few seconds up to multiple per second for a specific duration. Finally, *service failures*, in which events are generated by dependent applications, involved groups of randomly picked low-degree nodes, i.e., purposefully not

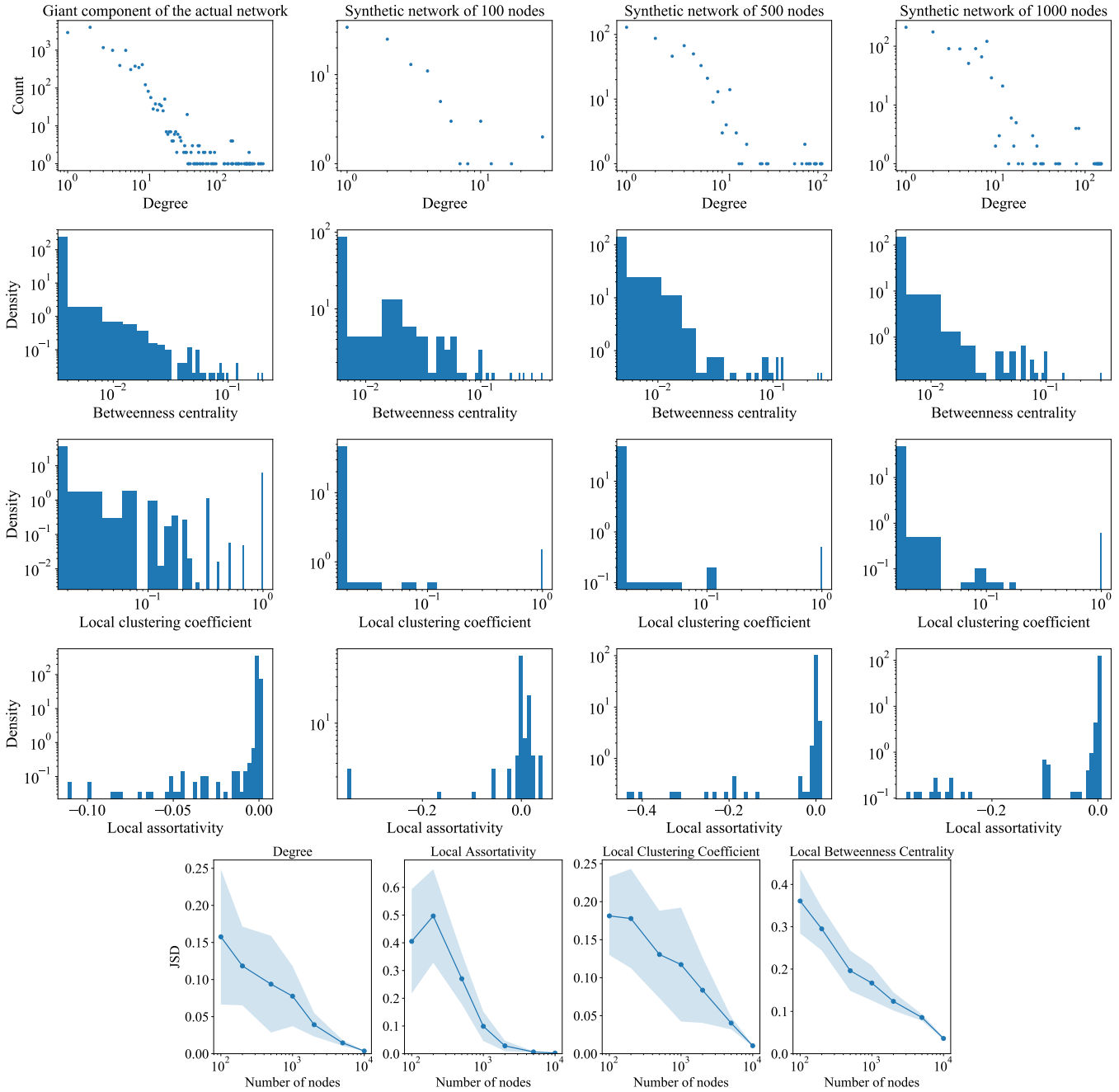

Fig. 2. (Top 4 rows) Distributions of key network metrics (degree, betweenness centrality, local clustering, local assortativity) for the giant component of the actual network (Dataset 1) (first column) and various sizes of synthetic structural connectivities generated using the process described in Section III-A2. Unsurprisingly, differences in distributions decrease as the size of the synthetic network grows (bottom row). Nevertheless, there is reasonable agreement for networks as small as 100 nodes which were needed to accommodate comparison with state of the art techniques in Sections III-D and III-E.

linked to any feature of the underlying structural connectivity. Here, failures consisted of bursts of events generated over a short time-span of up to a few seconds and occurring at random intervals.

In all four cases, temporal changes in functional connectivity were controlled by a parameter determining the probability of a functional connectivity starting/stopping on a daily basis.

The second class of event-emitting processes produced events occurring at random times on randomly chosen nodes (background noise). The rates of such events for each node were set so that the distribution of the number of events per node (over the entire synthetic dataset) roughly followed that of the real dataset (modelled for simplicity as a power law distribution with exponent $\alpha = 1.8$). Figure 3 provides a comparison of both distributions, along with a comparison of the number of events emitted per day. It can be observed that the daily rates for the synthetic data show some burstiness although less pronounced than in the actual dataset.

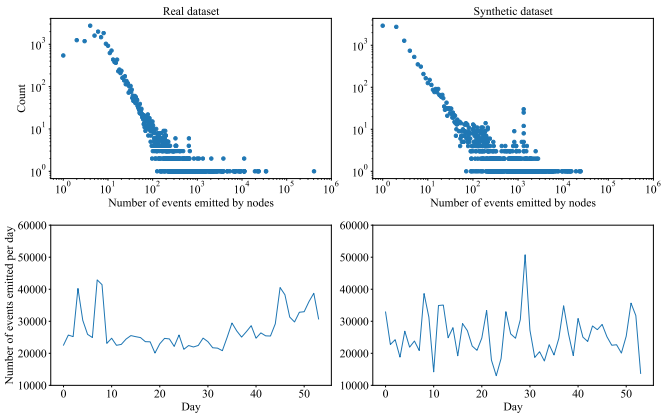

Fig. 3. Distribution of the number of events per node (top), distribution of the cumulated number of events emitted per day (bottom) for the real (left) and synthetic (right) datasets.

B. Validation: Pseudo-ground truth

Acquiring ground truth for evaluating our method is extremely difficult from a practical point of view, as this would require being able to label each device in the network with the different functional connectivities it is a member of, at all times. In the following, we use the type of events a device emits as the proxy for assessing our method in the absence of (absolute) ground truth. The intuition behind this assumption comes from our previous work [2], where we extensively studied Dataset 2 and discovered that only 0.5% of all devices emit events of more than one type and that devices seem to be emitting events that pertain to their functionality in the network. Types refer to the functionality of the running service emitting the event; examples of event types include 'router', 'LANSwitch', 'JVM', 'Linux', 'NetApp', 'VMWare', indicating functionality related to routing and switching, virtualised servers, and storage.

In order to quantify our intuition, we define an event-type distance metric on the types of events emitted by pairs of nodes. Specifically, if devices i and j emit events of types $\{t_1, t_2, ..., t_n\}$ in proportion $\{p_1^i, p_2^i, ..., p_n^i\}$ and $\{p_1^j, p_2^j, ..., p_n^j\}$ such that $\sum_{k=1}^{n} p_k^i = \sum_{k=1}^{n} p_k^j = 1$, then we define their event-type distance as follows:

$$d = \frac{1}{2} \sum_{k=1}^{n_i} (p_k^j - p_k^i)^2 \tag{7}$$

Based on our findings from [2], we expect that nodes belonging to the same functional connectivity will have low values of d, compared to nodes that are not functionally connected. We ran our method with Dataset 2. More specifically, we used 85% of the recording (130 days) to train our method (i.e. optimising the free parameters) and the remaining 15% (21 days) to compute the functional connectivity. To maintain consistency with the method of Kobayashi et al. [3], the window (i.e. the unit of adaptation time, i.e., when probabilities are updated) was set to 1 day. The maximum delay τ_{max} over which cross-correlations were calculated was set to 120s. This is consistent with the values used in the subsequent sections. Figure 1 shows the event-type distance d for different values of the threshold on the edge probability. We observe that for all values of the edge probability threshold, the calculated average distance for all pairs of nodes connected in the functional topology output by our method is at least half the average type distance for events emitted by devices not connected in the produced functional connectivity. For comparison purposes, we also provide the distance metric when considering structural links (orange dashed line). It is evident by the average type distance, that physically connected nodes rarely emit events of the same type. We note that, as the threshold value increases, the distance appears to decrease, in line with our intuition; as the criterion for identifying an edge in the functional topology gets stricter, nodes inferred to be functionally connected appear increasingly more likely to emit events of the same type.

Analysis of the functional connectivity. For all sensible values of the edge probability threshold, our method produces a graph that consists of a giant component and a number of smaller connected components (each one consisting of 10 or fewer devices). The distribution of component sizes, along with the number of events emitted by devices belonging to these components, is shown in Figure 4. The giant component consists of more than 80% of the network nodes. It encompasses multiple functional connectivities which, as we demonstrate below, are easy to cluster and retrieve. Figure 5(left) illustrates the graph that our method produced (with the parameters described above) for a threshold value of 0.8. Each colour identifies a unique event type, as depicted in Figure 6. Whilst nodes are coloured according to their most frequently emitted event type, edges are coloured according to the types of their end-points if those are identical, black otherwise. Note that more than one clusters of the same colour exist; i.e., the method can identify distinct functional connectivities from the time series of emitted events even when these events are of the same type. For example, two different and independent virtualised server deployments may be emitting the same type of event (e.g., 'VMWare'), even though they belong to separate functional connectivities. The fact that the various clusters identified in the graph appear to be fairly homogeneous in colour suggests that our method is indeed able to identify functional connectivities (provided that event types are an appropriate surrogate of functional connectivity, as suggested by our analysis of the data [2].

To better understand the relationship of functional connectivities and types of emitted events, we used the Louvain community detection algorithm [11] to identify sub-components in the giant component. We also included the small connected components identified by our method (which we assume to be independent functional connectivities of their own). In Figure 7 we illustrate the proportion of the most common type in each functional connectivity, as a function of the edge probability threshold. If a functional connectivity c consists of $|c|$ nodes that emit events of types $\{t_1, t_2, ..., t_n\}$ in proportion $\{p_1^c, p_2^c, ..., p_n^c\}$, the proportion of the most commonly emitted type is given by the weighted sum of the respective proportion for each functional connectivity:

$$prop = \frac{\sum_c |c| \times max_{1 \leqslant i \leqslant n}(p_i^c)}{\sum_c |c|} \tag{8}$$

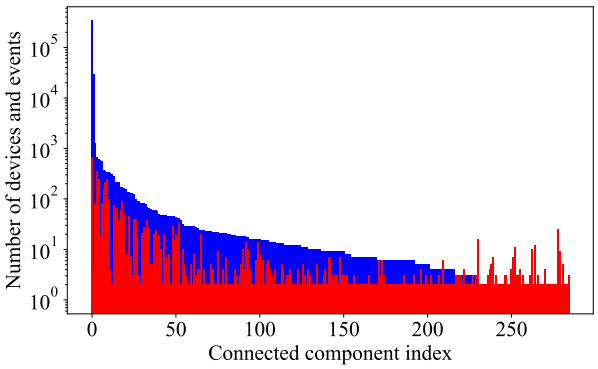

Fig. 4. Number of devices per connected component (red) and events they emitted over the last 3 weeks of recorded data (blue). Only components of more than one node are displayed. The probability threshold is set to 0.8.

First, we observe that the proportion of the most common type across all connectivities is significantly higher than that when ignoring functional connectivities, confirming that our method does identify groups of nodes that (mostly) emit events of a single type (we remind the reader that event type is not part of the information used to infer connectivity, only event times are). Second, this proportion increases with the threshold, which indicates that the precision of the method increases with the threshold; we extensively investigate the performance of our method with respect to precision and sensitivity in Section III-C.

Comparison to the structural connectivity. We have shown that our method has a high propensity to identify groups of nodes that emit events of the same type. Based on our analysis of the data, we believe these groups to be functional connectivities. Here, we show that knowledge of the event types (which, as a reminder, is not used in our inference method) and that of the structural connectivity would not have permitted to extract said functional connectivities. To illustrate this, we once again plot the inferred functional connectivities of Figure 5(left) but replacing the functional edges by the known structural edges (using the same colourings scheme). The result is depicted in Figure 5(right).

This analysis reveals that the functional connectivities inferred by our method cannot be predicted based on structural connectivity. Indeed, functionally connected nodes feature virtually no physical connections between them. Further, we observe that most links are black (indeed, 5150 out of 5207 are black), which show that they connect nodes emitting events of different type. Finally, the graph reveals the presence of hubs, i.e., very small (typically less than 5 nodes) but highly dis-assortative functional connectivities originating thick bundles of connections to various other functional connectivities. Careful examination (it will be helpful to the reader to use the zoomable version of the Figure) reveals that these hubs emit events of type 'NetApp', possibly reflecting data storage devices supporting back-end services. This suggests that the method could provide the means to assist with root-cause analysis.

	Condition Positive		Condition Negative	
PCP*	TP	$p_e(t_w) > 0.5$ $s_e(t_w + 1) > 0$	FP	$p_e(t_w) > 0.5$ $s_e(t_w + 1) \leqslant 0$
PCN*	FN	$p_e(t_w) \leqslant 0.5$ $s_e(t_w + 1) > 0$	TN	$p_e(t_w) \leqslant 0.5$ $s_e(t_w + 1) \leqslant 0$
NPC*	One node does not emit event at $t_w + 1$			

TABLE I
CONFUSION MATRIX. *PCP = PREDICTED CONDITION POSITIVE, PCN = PREDICTED CONDITION NEGATIVE, NPC = NO PREDICTED CONDITION

C. Validation: Predictive power

To further assess the performance of our method, we consider a more operational perspective on the usefulness of the concept of functional connectivity, namely, predictive power, whereby the inference of a functional link between two nodes enables us to make a statement about the likelihood that if events occur at one of the two nodes, events are also likely to occur at the other node. We first consider **Dataset 1** and investigate the predictive power of the method when systematically varying the length of the data over which the method is trained (from 2 to 50 days). The testing period (unseen data) consisted of the first 2 days of data after the training period. To maintain consistency with the method of Kobayashi et al., the window (the unit of adaptation time, i.e., when probabilities are updated) was set to 1 day. For this experiment (and all further experiments unless stated otherwise) the threshold (determining whether an edge existed) was set to 0.5. This is fairly arbitrary, and, as we will discuss in Section V, not necessarily helpful. As evidenced by the bottom panel of Figure 8, whilst selecting a low threshold does increase sensitivity, the gain is small in comparison to the loss in precision (almost a factor 2 between precisions at thresholds 1 and 0.5). The maximum delay τ_{max} over which cross-correlations were calculated was set to 120s. Again, this value was chosen to facilitate comparison with Kobayashi et al. as it corresponds to 2 bins of 1 minute. To quantify predictive power in the absence of ground truth, we adopted the following definitions (summarised in Table I). An edge is a true positive if the method predicted an edge and there was short-lagged activity across this edge in the testing period such that the score would predict the presence of an edge. An edge is a false positive if the method predicted an edge and there was some short-lagged activity across this edge in the testing period but the score for this activity would not predict the presence of an edge. True and false negatives are defined as the logical counterparts of true and false positives. It is essential to note that these definitions are contingent to short-lagged interaction happening in the testing period. This is because lack of activity across an edge over a period does not provide any information as to the existence of an edge. The edge might exist but not be active over the period. Such property was explicitly included in the construction of the model (see Section II-B).

Figure 8(top) shows the evolution of precision and sensitivity as the length of the training set was varied between 2 and 50 days. Whilst sensitivity remains stable for most length of training data, precision shows a gradual drop as the length of training data increases. This could suggest that the underlying functional topology changed during the record (this will be

Fig. 5. (Left) Functional topology of the graph for a probability threshold of 0.8. (Right) Structural topology using the mapping of the functional connectivity on the left. Detailed, zoomable versions of those figures are available from https://figshare.com/s/7cbfda9df3222e37710e and https://figshare.com/s/9b59a8f1ef882124700e, respectively.

Fig. 6. Mapping of the type to a specific colour

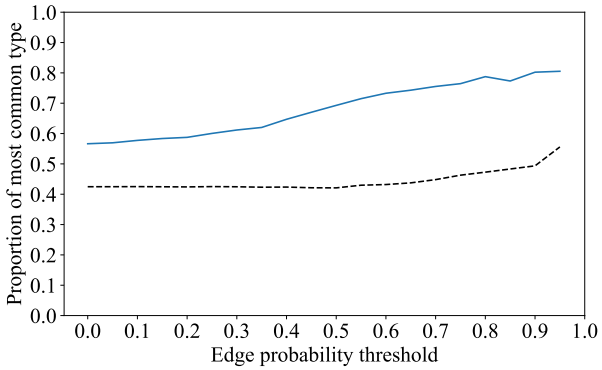

Fig. 7. Proportion of the most common type in each connectivity (as defined in Equation 8) as a function of the edge probability threshold. The black dotted line denotes the proportion of the most common type found across all connected nodes irrespective of community membership.

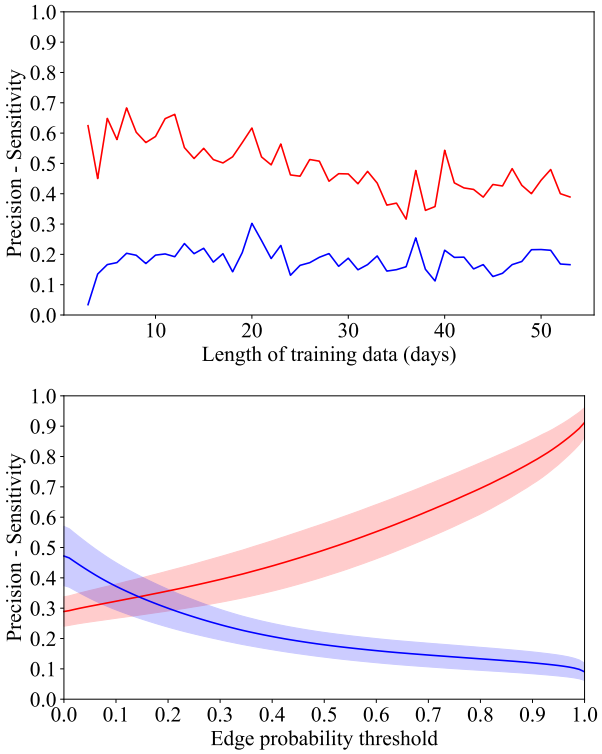

Fig. 8. (Top) Precision (red) and sensitivity (blue) as a function of the length of the training data (single run, full dataset). (Bottom) Average precision (red) and sensitivity (blue) as a function of the choice of edge probability threshold.

investigated below). To provide more confidence into the result, we repeated the experiment and averaged performance over 50 subnetworks of 1000 nodes picked at random.

As shown by Figure 9, sensitivity once again showed little sensitivity to the length of training data. Precision was slightly higher, and interestingly, there was less evidence of the decay seen when the full dataset was used. This is a somewhat counter-intuitive observation at first but can be explained in terms of the (limited) ability of a single (small) set of hyper-parameters to model heterogeneity in different components of the underlying functional topology. By considering subnetworks, the amount of per-network heterogeneity is potentially reduced, which may compensate for the potential loss of precision due to a changing underlying connectivity. An interesting operational implication could be that in a highly

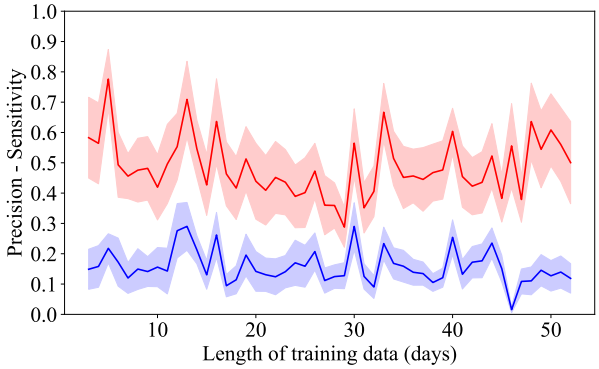

Fig. 9. Precision (red) and sensitivity (blue) as a function of the length of the training data (averaged over 50 subnetworks of 1000 nodes each).

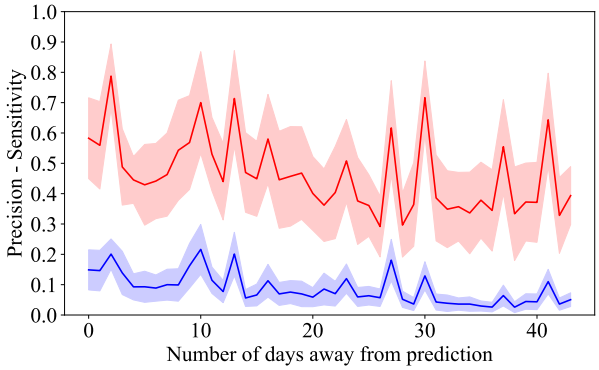

Fig. 10. Precision (red) and sensitivity (blue) as a function of the time (in days) between training data (first 10 days) and testing data (1 day), averaged over 50 subnetworks of 1000 nodes each.

heterogeneous environment, it might be beneficial to deploy multiple instances of the method (each dealing with specific types of events) rather than one.

To shed light on whether the underlying functional topology might have changed (in the absence of ground truth, this is difficult to establish) we analysed the method's predictive power when training was done over 10 days and the testing horizon was systematically varied between 1 and 40 days away from the training data. Figure 10 shows some evidence of gradual decline in both precision and sensitivity, suggesting there might have been changes.

To provide some insights into the behaviour of the method in response to changes in the underlying functional topology, we used synthetic data and systematically varied a parameter controlling the amount of changes in the set of functional connectivities involved on each day of the record (specifically, the probability that a functional connectivity starts/stops being active from one day to the other). Fifty networks of 1000 nodes were used, with an average of 20% of the functional connectivities described in Section III-A2 active at all times. Figure 11 shows that whilst performance remains approximately stable in the absence of changes (the blue line has no slope), suggesting the ability of the method to stabilise its predictions after 10 days, there is a steady drop in performance in the presence of changes, and the drop correlates with the amount of change unsurprisingly.

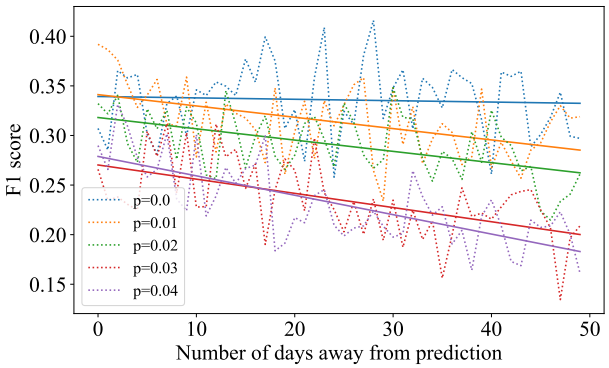

Fig. 11. F1 score (based on precision and sensitivity as defined in text) as a function of the time (in days) between training data (first 10 days) and testing data (1 day), averaged over 50 subnetworks of 1000 nodes each. Linear regressions are fitted to highlight the trends.

D. Comparison with Kobayashi et al. [3]

The method by Kobayashi et al. [3] assumes a direct acyclic graph (DAG) of events corresponding to the causality of events and proceeds in three steps. First they preprocess the data and remove events of time series that show strong temporal periodicity. Then, for every pair of nodes (X,Y), they state that an edge is not formed if there exists at least one node Z, such that nodes X and Y are conditionally independent ($P(X,Y|Z) \approx P(X|Z)P(Y|Z)$). Independence is tested using the conditional cross-entropy and the G-square test:

$$G^2 = 2mCE(X,Y|Z),$$

where m is the duration of the recording. Finally, they post-process the data and remove frequently appearing edges to enable the detection of unusually important causality. Our results were obtained using the authors' implementation available at https://github.com/cpflat/LogCausalAnalysis.

A comparison between their method and ours is challenging for three main reasons: (1) their method outputs DAGs denoting causal relationships between events based on activity taking place over a day, whereas our method infers an undirected functional topology based on activity taking place over a chosen amount of time; (2) their method requires event descriptors; (3) its greater time complexity (see Section III-F) makes it very impractical to deploy on the kind of large datasets for which our method is designed.

In principle we could apply the same experimental schedule as in Section III-C, however, we found that unlike with our method, the daily networks inferred were changing substantially (e.g., from 23% overlap on consecutive days to 5% over 2 days). The reason for this was found to be the low density of events such that events occurring on the day were not necessarily representative of events occurring on a different day. This means that training the method over 10 days or over the last of these 10 days would yield the same set of DAGs, thus making a comparison with our method unfair. Since only sensitivity is affected by this property, when using real data, we only report precision.

Because Kobayashi et al.'s method requires event descriptors, we used **Dataset 2**. Both methods were trained over an increasing number of days and performance was measured

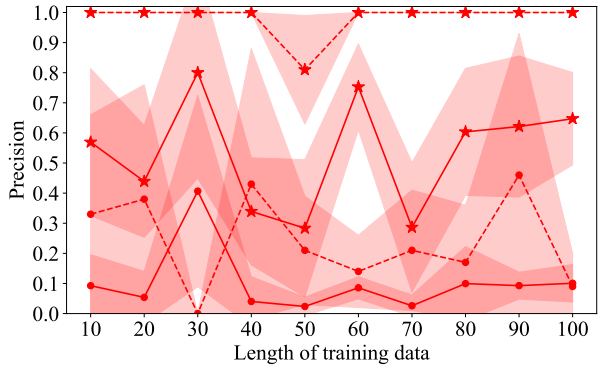

Fig. 12. Precision using our (star) and Kobayashi's (circle) scoring method for the method of Kobayashi et al. (dashed line) and ours (solid line). Data correspond to mean and standard deviation over the 10 days of unseen data.

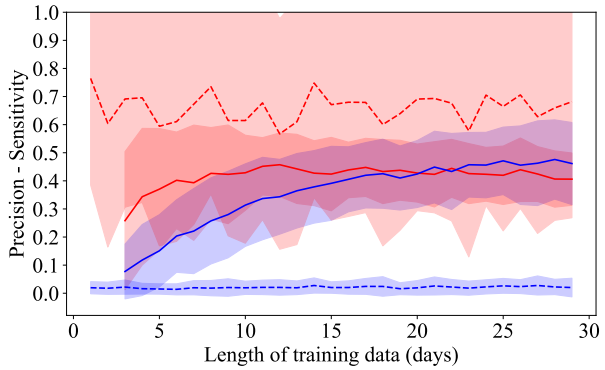

Fig. 13. Precision (red) and sensitivity (blue) for our method (solid line) and Kobayashi et al.'s method (dashed line). Length of training data is varied between 3 and 29 days. Performance is measured against ground truth on the next day.

over the next 10 days of unseen data. Since Kobayashi et al. predict causal interactions whilst our method returns undirected networks, precision was calculated using both criteria. Figure 12 shows that irrespective of the performance measure used, the method by Kobayashi et al. yields significantly higher precision. However, this must be put in context of a huge discrepancy in the number of events being predicted. Whereas our method returned over thousands of functional edges on a daily basis (up to over 100,000 for the longest training periods), the method by Kobayashi et al. only rarely returned more than 10 events per day (2.6%) and often (80%) none at all with an average of 1.2 edges predicted per day. Below, we will use synthetic data to make this case more fully.

The lack of stability in day-to-day inference as well as the low number of predicted events returned by the method of Kobayashi et al. can be attributed to the low density of events per node per day in the dataset (0.5). Experiments (results not shown) revealed that with higher densities, the percentage of overlap between day-to-day predictions increases (up to 30% for 100 events per node per day) albeit far inferior to that of our method (>90% overlap for densities from 0.7 to 100 events per node per day) and at the cost of much longer computations (both our method and that of Kobayashi et al. have a dependence on the number of events to be considered). In what follows, we used synthetic data with a sufficiently high density of events to provide both precision

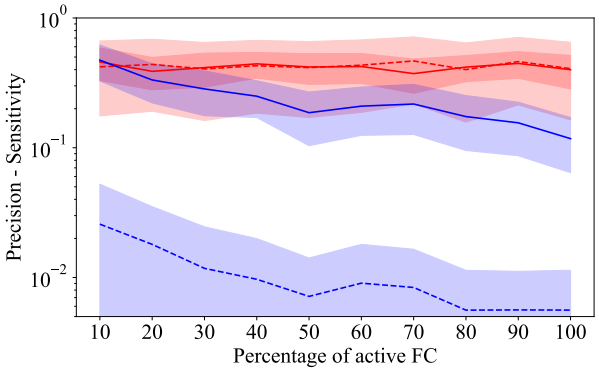

Fig. 14. Precision (red) and sensitivity (blue) for our method (solid line) and Kobayashi et al.'s method (dashed line) when the percentage of active functional connectivities os varied between 10% and 100%, all other parameters being equal. To make a straightforward comparison possible, our method was configured to return the same precision as that of Kobayashi et al., thus the near perfect overlap between red curves.

and sensitivity. The testing protocol considered is equivalent to that in Section III-C but using static networks of 300 nodes over 30 days (to reduce computational cost). Figure 13 shows that whilst the precision of our method is somewhat inferior to that of Kobayashi's, our sensitivity is far superior. Based on our earlier observation that the choice of threshold 0.5 whilst slightly improving sensitivity, dramatically reduces precision, we also calculated precision and sensitivity for a threshold of 0.9. This led to 20% improvement in precision (still less than Kobayashi) and a 33% drop in sensitivity (results not shown).

It will be noted that we used a very low density of functional edges. As illustrated in Figure 14, this is because in the method of Kobayashi et al., the likelihood of an edge existing relies on a p-value derived from conditional cross-entropies. The higher the density of functional edges, the more likely it is to find a node Z such that there is conditional independence between X and Y. There is also a drop in sensitivity for our method. This is because the number of pairs showing significantly higher scores than those picked at random is dropping.

E. Comparison with Hallac et al. [12]

Hallac et al. [12] extended the graphical Lasso algorithm and developed a method to solve for $\Theta = (\Theta_1, \Theta_2, ..., \Theta_T)$ a set of symmetric positive definite matrices:

$$min_{\Theta \in S^p_{++}} \sum_{i=1}^{T} -l_i(\Theta_i) + \lambda||\Theta_i|| + \beta \sum_{i=2}^{T} \Phi(\Theta_i - \Theta_{i-1}),$$

where T is the number of windows, $l_i(\Theta_i) = n_i(\log \det\Theta_i - Tr(S_i\Theta_i))$ is a function that encourages Θ_i to be close to S_i^{-1} the inverse of the empirical covariance (if S_i is invertible), n_i is the number of observations, $||\Theta_i||$ is the semi-norm of Θ_i, λ is a positive constant that is adjusted to enforce the sparsity of the covariance matrix, $\Phi(\Theta_i - \Theta_{i-1})$ is a convex penalty function minimised at $\Phi(0)$, which encourages similarity between Θ_t and Θ_{t-1} and β is a positive constant determining how strongly correlated neighbouring covariance estimations should be. The connectivity at time t is then simply extracted from the non-zeros values of the inverse of the precision matrix Θ_t.

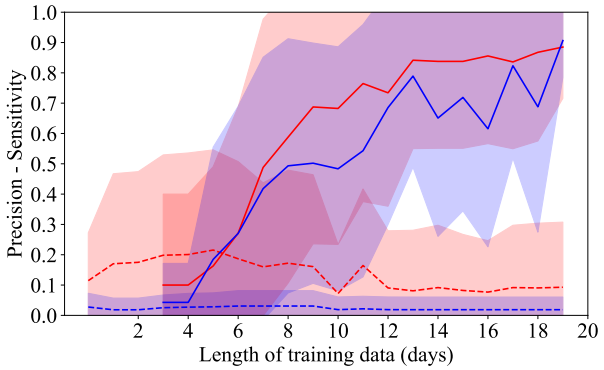

Fig. 15. Precision (red) and sensitivity (blue) for our method (solid line) and that of Hallac et al. (dashed line) when length of training data is varied between 1 (3 for our method) and 20 days. Performance was calculated as the mean over the 10 days of unseen data and was averaged over 10 networks of 50 nodes each. On average 20% of the functional connectivities were active.

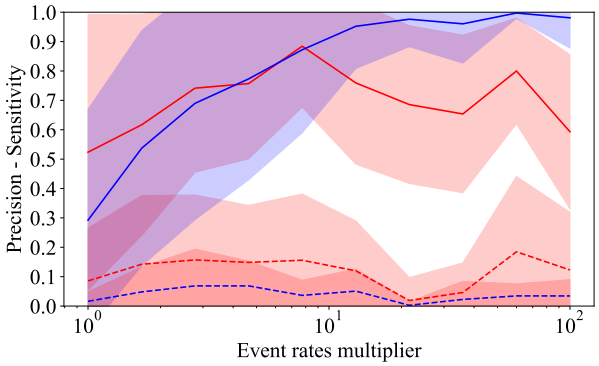

Fig. 16. Precision (red) and sensitivity (blue) for our method (solid line) and Hallac et al.'s method (dashed line) when the density of events is varied between 1 and 10,000 events per day (for networks of 100 nodes). Performance is measured against ground truth.

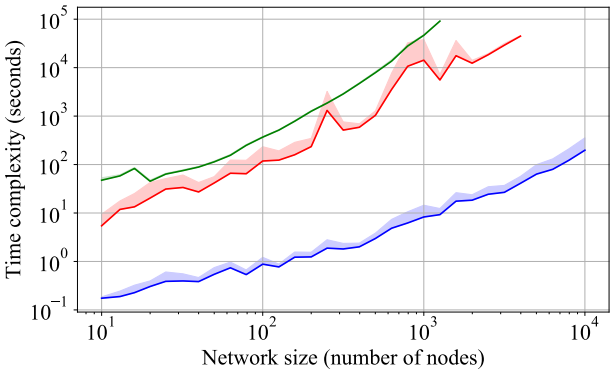

Fig. 17. Time complexity (in seconds) for our method (blue), Hallac et al. (green) and Kobayashi et al. (red) when network size is varied between 10 and up to 10,000 nodes (depending on methods).

Our results were obtained using the authors' implementation available at https://github.com/davidhallac/TVGL.

The challenge of providing a comparison is in setting a suitable criterion for setting the various parameters of Hallac et al.'s method, most critically, number of windows, bin size, choice of penalty function and parameters λ and β. Since the number of windows is an arbitrary choice, we set it to 1 day, as in Kobayashi et al. The penalty function was set to the Laplacian because smooth changes are assumed to take place in the real data. All other parameters were subjected to grid search to maximise the resulting F1 score. In the absence of ground truth, predictive power was assessed in terms of the method's ability to predict an event. This is substantially different from any of the tests used previously but is fair, if not particularly favourable to either method.

The presence of an edge in Hallac et al.'s method was assessed on the basis of non-zero values of the inverse of the Θ matrix it returned. As shown in Figure 15, our method outperforms that of Hallac et al. for all configurations considered. In particular, because the networks are so small (size chosen due to the poor time complexity of the method of Hallac et al.), our method can achieve high precision and sensitivity after only 10 days of training data. In a final experiment, we examined whether the density of events (within the limit of what was computationally possible) could explain the poor performance of Hallac et al's method. We systematically varied the density of events through multiplying the functional event rates to reduce the sparsity of the binned matrix used by Hallac. As shown in Figure 16, whilst the increased density did result in a higher sensitivity for our method, both precision and sensitivity for Hallac et al. remained very low.

F. Scalability Analysis

A major challenge in this work was the inability of the benchmark methods to handle the size of the real datasets considered here. In this Section, we provide a comparative analysis of how each method scales with network size. For the purpose of this analysis, we ignored any consideration of performance but focused on measuring time complexity when all three methods were set to operate on an as similar setup as possible. Concretely, we used synthetic data and simulated activity over 10 days. Window size was set to 1 day in all three methods. Bin size was 1h for Hallac et al.s' method. Cross-correlations were calculated up to 2 minutes in our method.

Figure 17 demonstrates the clear superiority of our method when the number of nodes in the network is systematically varied, with roughly a factor 10^4 for network sizes of 10^3 nodes. No simulations were carried out for network sizes greater than 10^3 nodes for the benchmark methods due to the excessive time it would have taken to do so.

IV. Related Work

A number of approaches that are based on traditional data mining techniques have been proposed to extract useful information about the operation of networks and services that could be used for real-time [13], [14] and post-incident analysis [15], [16] and for understanding performance problems [17]. Recently, the use of supervised and unsupervised learning for detecting anomalies in network and service log data has been explored (see [18] for a summary). Workflow construction through processing of collected log data [4], [5] and filtering of unimportant network nodes based on the notion of graph vertex entropy [8] and supervised machine learning [19] have also been proposed. Our method is complementary to the aforementioned algorithms and systems, which can be more efficient and effective when applied to smaller datasets of

collected network log data known to have been produced by network devices and servers belonging to the same functional topology, i.e., collectively providing a specific service or core network functionality.

Our work is inspired by research conducted in the context of other complex networks. In brain networks, even the most advanced forms of imaging cannot provide an accurate or complete description of structural connectivity, see [20] for example. Although link prediction methods are being developed to attempt to extract missing information [21], [22], a more promising approach is to infer connectivity from the temporal evolution of events occurring at node level [23] without assuming any prior knowledge regarding connectivity. In brain neural networks, for example, network inference may rely on spike dynamics [24]. In gene regulation, published methods primarily use gene expression data derived from micro-arrays [25].

A number of methods have been developed to infer connectivity changes in a network by adapting Bayesian network methods [26]–[29]. These methods assume a model of event propagation. At their core is the belief that the current state can be predicted, with some probability, based on the previous state or states. In the context of very sparse data, such as in computer network and service deployments, when an event in A may trigger an event in B in only a small percentage of the cases, such an assumption is problematic. Further, and as should be apparent from the failure scenarios we discussed in the context of generating the synthetic data, there can be great heterogeneity in how event times 'follow' from a root cause event. For example, when intelligent polling is used (whereby a monitoring server polls devices that might be affected by a failure elsewhere) can result in event times that are not intrinsic to the network infrastructure itself but rather depend on how the monitoring system is set up to react to the root cause (such information not being available to us). For this reason, methods such as the Markovian model used in temporal exponential graphs [30], various adaptations of the Kalman filter [31], [32] or methods relying on propagation of cascades [33] are unlikely to be as effective as a method that will solely rely on pair-wise information. In fact, the use and adaptation of pair-wise correlations is the basis of many methods that do not assume an event propagation model, e.g., [34]–[36]. However, these adaptations typically result in methods that do not scale well to large networks over long recordings.

A final class of methods relies on the estimation of a time-varying covariance matrix to encode the correlation structures at each observation, e.g., [37]–[39]. Constraints of sparsity (in the network of interdependencies between the nodes) are enforced by way of lasso penalty. These methods typically do not scale well to large examples, as we showed for Hallac et al. [12].

V. Conclusion

Monitoring, responding to, and predicting, failures in a large scale network deployment is a key responsibility of network operators. The sheer amount of data generated by devices makes this task particularly difficult. In this paper, we sought to present an alternative framework to representing and modelling network events. This framework is based on the concept of functional connectivity first introduced in neuroscience. In contrast to structural connectivity, the underpinning physical infrastructure, functional connectivity represents statistical dependences between the activities of nodes in the network. In this paper, we specifically focused on statistical dependence based on the amount of short-lagged interactions between them. We presented a new statistic to robustly assess the presence of statistical dependence between two nodes. By deploying this statistic in a windowed-fashion and embedding it into a predictive framework, we were able to develop an inference model of time-varying functional connectivity able to meet three key challenges: (a) non-stationarity of the underlying structural and functional connectivities, (b) sparsity of the time-series of events limiting the effectiveness of classical measures of statistical dependence, and (c) lack of information as to how events follow from root causes.

The fact that a substantial amount of this paper was focused on the methodological aspects of inferring functional connectivity in large-scale commercial deployments should not detract from its main focus, namely, making the case for functional connectivity as a powerful tool in the arsenal of network operators. Through our various validations effort, we have sought to demonstrate a number of benefits. First, as shown in Section III-B, the inference of functional connectivity provides the kind of insights that might otherwise require intensive processing of the content of the events. We showed that the extracted communities were characterised by (among other things) great homogeneity in the type of events that originating them. Such information can be available to operators but commercial experience (as well as academic papers such as [3]) shows that recovering it typically entails a significant cost because event descriptions can involve templates that change over time, manual entries that might be subject to human error, etc. Further, relying on event descriptions constrains the kind of functional relationships that will be identified to those that will have been anticipated by the operators setting up the system, e.g., services, expected failures. By relying on timestamps only, the concept of functional connectivity is agnostic to the origin of the dependencies. It tells the network operator that nodes that might not have been predicted to be related in any way, actually are, due to unexpected factors. Second, the concept of functional connectivity alleviates the need for accurate network discovery that afflicts most existing commercial systems. Currently fault localisation and root cause analysis depend on an accurate description of the network. This is challenging for two reasons. First, the network is complex, heterogeneous and changing. Second, automated discovery approaches cannot always capture the inherently multiplex nature of large-scale deployments. Instead, although functional dependencies are underpinned by structural connectivity, their identification is not contingent on having full and accurate knowledge of this structural connectivity. This was clearly illustrated in Section III-B, and Figure 5 particularly, showing the almost total lack of overlap between functional and structural connectivities. Indeed, we found an almost complete lack of structural links between nodes belonging to

a functional connectivity (as identified by community structure membership). Instead, structural edeges appear to link functional connectivities, highlighting their mediation but not causal role in the emergence of functional relationships. Thus, at the very least, one can think of functional connectivity as providing added value to current network discovery protocols. Finally, inference of functional connectivity, including *time-varying* functional connectivity, has useful predictive value for network operators. Even though functional connectivity is not effective (or causal) connectivity, a robust and comprehensive characterisation of correlation has long been established as an important step to *pinpoint the causes [of alarms] so that problems can be handled effectively* [40]. Functional connectivities can be thought of as spheres of influence such that in the presence of an incident, network operators (i) can rapidly discard events that might co-occur purely for spurious reasons, (ii) focus their effort on those nodes they know to be functionally related (including interpreting their content) and (iii) use this information to accelerate the process of fault localisation and root cause analysis. Importantly, inference of time-varying functional connectivity is crucial for dynamic computing environments where hardware, storage and network virtualisation enables the elastic provisioning of resources to a large and diverse set of services and applications. In such environments, service components (e.g. Docker containers or whole virtual machines) can be dynamically created, removed or migrated so that specific performance constraints are adhered to. We expect that functional connectivity inference will play a key role in fault analysis and prediction in such dynamic environments in the future.

Inferring functional connectivity is a hard problem. When using synthetic data for which ground truth was available, the F1-score only rarely exceeded 0.7 in the near-static case, 0.6 in the more dynamic case. However, this should not detract from the fact that the method was able to recover a substantial amount of the connectivity, including its changes over time, from an extremely limited amount of information. Indeed, it did so at least as well as state of the art methods in the near-static case, and usually better in the dynamic case. Importantly, unlike existing network inference methods (that typically do not handle sparse data well), it remains computationally tractable even with large networks (here, 10,000 nodes) over very long records (here, 10^7 observations). To be able to benchmark our method against state-of-the-art methods, we had to scale down to networks of size magnitudes smaller than our real-world application. The lack of scalability of these methods cannot be overstated.

Although our method produces weighted networks, where the weight denotes the strength of the interaction, in this paper, we have been thresholding those weights throughout, both for prediction and evaluation purposes. Use of a threshold has a number of disadvantages, from losing important information about high-confidence edges (their distribution and organisation) to giving the same importance to high- and low-confidence edges. It also potentially confers the inference with sensitivity to the choice of the threshold. Whilst our experiments did not show evidence of such sensitivity (at least in the scenario that was tested), our results did show that the marginal gain in sensitivity due to using a small threshold came at the cost of a substantial drop in precision. An alternative is to use a continuous loss function based on the probabilities returned by our model. We are currently developing a method for automatically inferring the best threshold.

Our method returns a graph. It is reasonable to ask whether there is any reason for it other than visualisation. Whilst the visualisation aspect cannot be underestimated in the context of a network management system, we see the graph representation as an essential, albeit yet to be fully explored, component of the concept of functional connectivity in the sense of it being a starting point for understanding and analysing the system. In this paper, we compared the properties of the (known) structural and (inferred) functional connectivities and were able to gain insights regarding the extent to which inferred functionality did capture an aspect of the events which was not included in the inference mechanism (namely, event types) as well as how structural connectivity underpinned connectivity between distinct functional connectivities. The operational implications of such insights remain to be seen. As we infer a time-varying connectivity, it will also be of interest to monitor how the characteristics of these inferred connectivities evolve over time. Indeed, it has been recently suggested that such analysis could help predict failures.

A key stumbling block in the development of this framework has been the absence of ground truth. Because functional connectivity only denotes statistical dependence (but not causal influence), there is no obvious way to provide an unequivocal assessment of how valid our inference is. In this work, we have used two approaches. One is to assess the extent to which the inferred model can predict future statistical dependences. Whilst this has operational value (e.g., filtering events based on knowledge that they are merely an expression of some latent statistical dependence to the activity of another node known to have emitted events), it is not an absolute measurement of quality since, for example, functional connectivity could be changing or nodes may not emit events during the period considered. Unsurprisingly, we have been consistently reporting low sensitivity values. The other approach is to use synthetic data. However, this has presented yet another challenge, namely, that of providing an accurate depiction of what happens in a live deployment. In this work, we have considered four types of network failure scenarios as proxy for functional connectivities. However, there are many other ways by which to define such connectivities. This highlights the need for controlled testbeds for experimentation. We are not aware of any and sadly there is little scope for experiments in commercial deployments, particularly when they involve critical services.

In conclusion, this is a first step toward developing the notion of dynamic functional connectivity inference in network management. To fulfil its full applicative potential, a more complete understanding of the various assumptions and parameters underpinning it must be obtained, which will be the subject of our future work.

ACKNOWLEDGEMENTS

This research was funded by Moogsoft Ltd. and describes patented features of its product. The authors are grateful to the anonymous reviewers for their very constructive and helpful comments.

REFERENCES

[1] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat, "Evolve or Die: High-Availability Design Principles Drawn from Googles Network Infrastructure," in *Proc. of ACM SIGCOMM*, 2016.

[2] A. Messager, G. Parisis, R. Harper, P. Tee, I. Z. Kiss, and L. Berthouze, "Network events in a large commercial network: What can we learn?" in *Proc. of IEEE/IFIP NOMS AnNet*, 2018.

[3] S. Kobayashi, K. Otomo, K. Fukuda, and H. Esaki, "Mining causality of network events in log data," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 53–67, 2018.

[4] X. Yu, P. Joshi, J. Xu, G. Jin, H. Zhang, and G. Jiang, "Cloudseer: Workflow monitoring of cloud infrastructures via interleaved logs," in *Proc. of ASPLOS*, 2016.

[5] I. Beschastnikh, Y. Brun, M. D. Ernst, and A. Krishnamurthy, "Inferring models of concurrent systems from logs of their behavior with csight," in *Proc. of ICSE*, 2014.

[6] R. Harper and P. Tee, "A method for temporal event correlation," in *NOMS 2019 - 2019 IEEE/IFIP Network Operations and Management Symposium*. IEEE, in press.

[7] M. Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Science of Computer Programming*, vol. 53, no. 2, pp. 165–194, 2004.

[8] P. Tee, G. Parisis, and I. Wakeman, "Vertex entropy as a critical node measure in network monitoring," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 646–660, 2017.

[9] S. Kätker and M. Paterok, "Fault isolation and event correlation for integrated fault management," in *Proc. of IFIP/IEEE IM*, 1997.

[10] A. Messager, N. Georgiou, and L. Berthouze, "A new method for the robust characterisation of pairwise statistical dependency between point processes," *arXiv:1904.04813v1*, Apr. 2019.

[11] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[12] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, "Network inference via the time-varying graphical lasso," *arXiv:1703.01958*, 2017.

[13] J. P. Rouillard, "Real-time log file analysis using the simple event correlator (sec)," in *Proc. of LISA*, 2004.

[14] K. Yamanishi and Y. Maruyama, "Dynamic syslog mining for network failure monitoring," in *Proc. of KDD*, 2005.

[15] A. Oprea, Z. Li, T. F. Yen, S. H. Chin, and S. Alrwais, "Detection of early-stage enterprise infection by mining large-scale log data," in *Proc. of DSN*, 2015.

[16] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda, "Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks," in *Proc. of ACSAC*, 2013.

[17] S. Roy, A. C. KÃűnig, I. Dvorkin, and M. Kumar, "Perfaugur: Robust diagnostics for performance anomalies in cloud services," in *Proc. of ICDE*, 2015.

[18] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *Proc. of ISSRE*, 2016.

[19] R. Harper and P. Tee, "The application of neural networks to predicting the root cause of service failures," in *Proc. of IFIP/IEEE IM AnNet*, 2017.

[20] C. Thomas, Q. Y. Frank, M. O. Irfanoglu, P. Modi, K. S. Saleem, D. A. Leopold, and C. Pierpaoli, "Anatomical accuracy of brain connections derived from diffusion mri tractography is inherently limited," *Proceedings of the National Academy of Sciences*, vol. 111, no. 46, 2014.

[21] Q. Liu, S. Tang, X. Zhang, X. Zhao, B. Y. Zhao, and H. Zheng, "Network growth and link prediction through an empirical lens," in *Proc. of ACM IMC*, 2016.

[22] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[23] I. Brugere, B. Gallagher, and T. Y. Berger-Wolf, "Network structure inference, a survey: Motivations, methods, and applications," *arXiv:1610.00782*, 2016.

[24] E. N. Brown, R. E. Kass, and P. P. Mitra, "Multiple neural spike train data analysis: state-of-the-art and future challenges," *Nature Neuroscience*, vol. 7, no. 5, pp. 456–461, 2004.

[25] M. Hecker, S. Lambeck, S. Toepfer, E. Van Someren, and R. Guthke, "Gene regulatory network inference: data integration in dynamic models?a review," *Biosystems*, vol. 96, no. 1, pp. 86–103, 2009.

[26] F. Dondelinger, S. Lèbre, and D. Husmeier, "Non-homogeneous dynamic bayesian networks with bayesian regularization for inferring gene regulatory networks with gradually time-varying structure," *Machine Learning*, vol. 90, no. 2, pp. 191–230, 2013.

[27] Z. Wang, E. E. Kuruoglu, X. Yang, Y. Xu, and T. S. Huang, "Time varying dynamic bayesian network for nonstationary events modeling and online inference," *IEEE Transactions on Signal Processing*, vol. 59, no. 4, pp. 1553–1568, 2011.

[28] J. W. Robinson and A. J. Hartemink, "Learning non-stationary dynamic bayesian networks," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3647–3680, 2010.

[29] L. Song, M. Kolar, and E. P. Xing, "Time-varying dynamic bayesian networks," in *Advances in Neural Information Processing Systems*, 2009, pp. 1732–1740.

[30] F. Guo, S. Hanneke, W. Fu, and E. P. Xing, "Recovering temporally rewiring networks: A model-based approach," in *Proc. of ICML*, 2007.

[31] V. Carluccio, N. Bouaynaya, G. Ditzler, and H. M. Fathallah-Shaykh, "The akron-kalman filter for tracking time-varying networks," in *Proc. of IEEE BHI*, 2017.

[32] J. Khan, N. Bouaynaya, and H. M. Fathallah-Shaykh, "Tracking of time-varying genomic regulatory networks with a lasso-kalman smoother," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2014, no. 1, p. 3, 2014.

[33] M. G. Rodriguez, J. Leskovec, D. Balduzzi, and B. Schölkopf, "Uncovering the structure and temporal dynamics of information propagation," *Network Science*, vol. 2, no. 1, pp. 26–65, 2014.

[34] A. J. Oliner, A. V. Kulkarni, and A. Aiken, "Using correlated surprise to infer shared influence," *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 191–200, 2010.

[35] A. A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao, "Towards automated performance diagnosis in a large iptv network," in *Proc. of SIGCOMM*, 2009.

[36] Z. Zheng, L. Yu, Z. Lan, and T. Jones, "3dimensionall root cause diagnosis via co-analysis," in *Proc. of ICAC*, 2012.

[37] R. P. Monti, P. Hellyer, D. Sharp, R. Leech, C. Anagnostopoulos, and G. Montana, "Estimating time-varying brain connectivity networks from functional mri time series," *NeuroImage*, vol. 103, pp. 427–443, 2014.

[38] E. C. Wit and A. Abbruzzo, "Inferring slowly-changing dynamic gene-regulatory networks," *BMC Bioinformatics*, vol. 16, no. 6, p. S5, 2015.

[39] S. Zhou, J. Lafferty, and L. Wasserman, "Time varying undirected graphs," *Machine Learning*, vol. 80, no. 2, pp. 295–319, 2010.

[40] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo, "A Coding Approach to Event Correlation," in *Integrated Network Management IV*, A. S. Sethi, Y. Raynaud, and F. Faure-Vincent, Eds. Boston, MA: Springer US, 1995, pp. 266–277.